Set 3

Assume the following statements when answering the following questions.

```
Location loc1 = new Location(4, 3);
Location loc2 = new Location(3, 4);
```

1. How would you access the row value for loc1?

Using the getRow() method of loc1, loc1.getRow().

2. What is the value of b after the following statement is executed?

```
boolean b = loc1.equals(loc2);
```

The value of b is false.

3. What is the value of loc3 after the following statement is executed?

```
Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
```

The value of loc3 is (4, 4)

4. What is the value of dir after the following statement is executed?

```
int dir = loc1.getDirectionToward(new Location(6, 5));
```

The value of dir is 135 degrees (Southeast)

5. How does the getAdjacentLocation method know which adjacent location to return?

The parameter of the getAajacentLocation method is the direction of adjacent location.

According to the direction, returns the closest compass direction from this location toward target.

Set 4

Assume that grid is the Grid object

1. How can you obtain a count of the objects in a grid? How can you obtain a count of the empty locations in a bounded grid?

Using the grid.getOccupiedLocation() method to obtain the location occupied by a object

in a grid. Then Using the grid.getOccupiedLocation().size() to get a count of the objects in the grid. The grid.getNumRows() * grid.getNumCols return the size of grid,

grid.getNumRows() * grid.getNumCols() – grid.getOccupiedLocatiion().size()

is the count of the empty location.

2. How can you check if location (10,10) is in a grid?

Using the isValid() method, for example:

boolean flag = grid.isValid(new Location(10, 10))

if flag is true, then location(10, 10) is valid, vice versa.

3. Grid contains method declarations, but no code is supplied in the methods. Why? Where can you find the implementations of these methods?

Since Grid is declared as a interface, in Java a interface do not implement the methods, the classses which implements the Grid interface implemets those methods. From the relationship figure, BoundedGrid and UnboundedGrid implements the Grid interface, so we can find the implementations of these methods in the UnboundedGrid and BoundedGrid.

4. All methods that return multiple objects return them in an ArrayList. Do you think it would be a better design to return the objects in an array? Explain your answer.

No, although it is easy to get a object using the [] in array than the get method in the ArrayList, array has to set the size when declared, but ArrayList does not and it change length.

Set 5

1. Name three properties of every actor.

Color , location, direction

2. When an actor is constructed, what is its direction and color?

The color is blue and the direction is north.

3. Why do you think that the Actor class was created as a class instead of an interface?

Since the Flower , Bug and Rock have some same properties, they all are a actor in the grid. As a class, the Flower, Bug and Rock class can extends the Actor class, the relationship between Flower, Bug, Rock and Actor is "is a", but if created as a interface, the class implements the interface just use it, and not allow to create instance variable.

4. Can an actor put itself into a grid twice without first removing itself? Can an actor remove itself from a grid twice? Can an actor be placed into a grid, remove itself, and then put itself back? Try it out. What happens?

No, an actor can not put itself into a grid twice, it will raise an illegalStateException();

No, an actor can not remove itself from the a grid twice, it will raise an illegalStateException();

Yes, an actor can put itself into grid then remove itself and put itself into grid again.

5. How can an actor turn 90 degrees to the right?

Using the getDirection() and setDirection() methods, setDirectio(getDirection() + 90) ;

Set 6

1. Which statement(s) in the canMove method ensures that a bug does not try to move out of its grid?

```
 Location next = loc.getAdjacentLocation(getDirection());
  if (!gr.isValid(next))
        return false;
```

the statements ensure that a bug does not try ot move onut of its gird

2. Which statement(s) in the canMove method determines that a bug will not walk into a rock

```
 Actor neighbor = gr.get(next);
 return (neighbor == null) || (neighbor instanceof Flower);
```

the statements determine that a bug will not walk into a rock.

3. Which methods of the Grid interface are invoked by the canMove method and why?

The isValid() and get() methods of Grid interface are invoked by the canMove() method, Because it need check the next location is valid and whether object in the location can be replaced by the bug.

4. Which method of the Location class is invoked by the canMove method and why?

The method getAdjacentLoacation() of the Location class is invoked by thee canMove() method, Because before the check the next location, the grid has to obtain the next location.

5. Which methods inherited from the Actor class are invoked in the canMove method?

The getLocation(), getDirection() and getGrid() methods.

6. What happens in the move method when the location immediately in front of the bug is out of the grid?

Using the removeSelfFromGrid() to remove the bug from the grid.

7. Is the variable loc needed in the move method, or could it be avoided by calling getLocation() multiple times?

Yes, the variable loc is needed, and it could not avoided by calling getLocation() method. Since after the bug leaved, the grid has to put the flower on the original location.

8. Why do you think the flowers that are dropped by a bug have the same color as the bug?

Because when create the flower, it call the getColor() method of bug to set the flower's color.
9. When a bug removes itself from the grid, will it place a flower into its previous location?

If the bug removes itself from the grid in the move() method, it will place a flower into its previous location.

If the bug remove itself from the grid calling the removeSelfFromGrid() method, it would not place a flower in the previous location.

10. Which statement(s) in the move method places the flower into the grid at the bug's previous location?

```
 Location loc = getLocation();
 Flower flower = new Flower(getColor());
 flower.putSelfInGrid(gr, loc);
```

11. If a bug needs to turn 180 degrees, how many times should it call the turn method?

It should call 4 times the turn method.